Ghostscript Status
OpenPrinting summit April 2012

Michael Vrhel, Ph.D.

Artifex Software Inc.

San Rafael CA

Outline

Ghostscript overview

What is new and what is coming…

Color architecture

The Basics

Ghostscript is a document conversion and rendering engine.

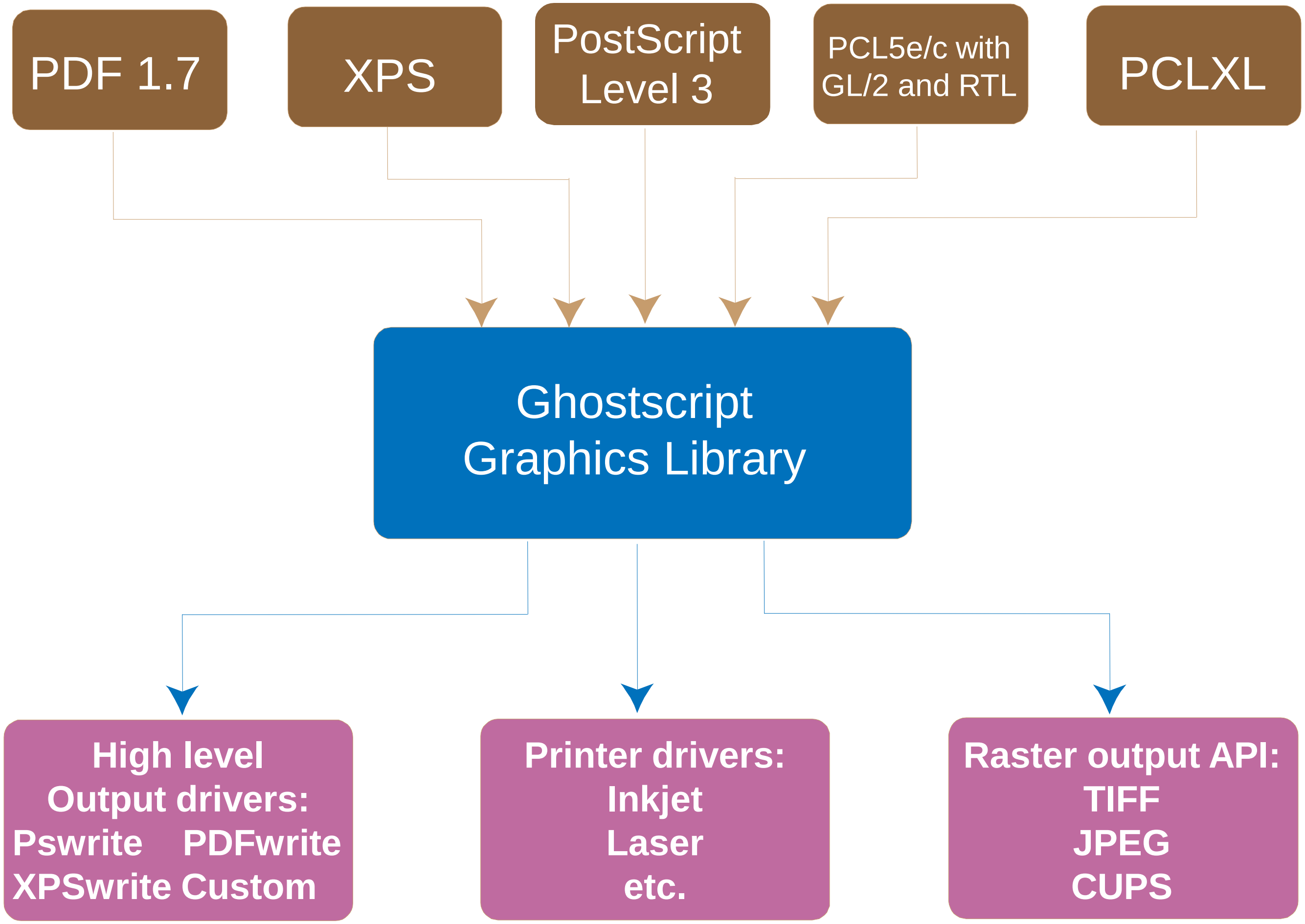
Written in C ANSI 1989 standard (ANS X3.159-1989)

Essential component of the Linux printing pipeline.

Dual GPL/Proprietary licensed. Artifex owns the copyright.

Source and documentation available at www.ghostscript.com

Graphical Overview


PDF 1.7

XPS

PostScript Level 3

PCL5e/c with GL/2 and RTL

PCLXL

Ghostscript Graphics Library

High level Output drivers: Pswrite PDFwrite XPSwrite Custom

Printer drivers: Inkjet Laser etc.

Raster output API: TIFF JPEG CUPS

Devices

Understanding devices is a major key to understanding ghostscript.

Devices can have high-level functionality. e.g. pdfwrite can handle text, images, patterns, shading, fills, strokes and transparency directly.

Devices may be set up to handle only certain high-level operations.

Graphics library has “default” operations. e.g. text
turns into bitmaps, images decomposed into rectangles.

In embedded environments, calls into hardware can be made.

Raster devices require the graphics library to do all the rendering.

Relevant Changes to GS since last meeting….

High speed halftoning (SIMD SSE) CMYK planar devices. (9.04)

Support for anti-aliasing when source contains transparency (9.04)

Re-enable of x11alpha as default device on Unix systems (9.04)

Object based color rendering (9.04)

Improved Black control in CMYK output (9.04)

Relevant Changes to GS since last meeting….

Source ICC Profile / Rendering intent override (9.04)

Git source control (9.04)

Experimental text extraction device released (9.04)

Now ships with littleCMS 2.3 (9.05)

Support for Proofing ICC Profiles (9.05)

Relevant Changes to GS since last meeting….

Support for Device Link ICC Profile (9.05)

Support for unmanaged color (9.05)

Embedding of ICC profiles in the TIFF, JPEG and PNG output devices (9.05)

Font set distributed with Ghostscript changed to the standard 35 Postscript-compatible fonts distributed by URW (9.05)

Includes modified OpenJPEG sources for JPEG2000 decoding (9.05)

Upcoming Changes to GS (release 9.06*)

Support for Output Rendering Intent. (in trunk)

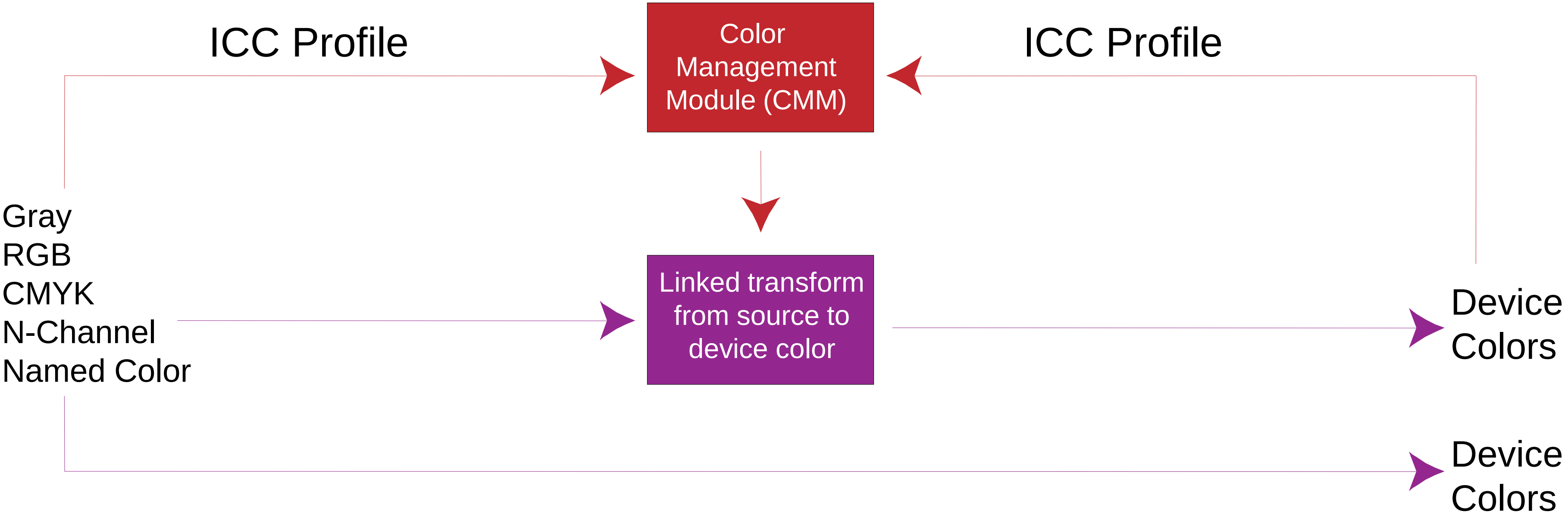
Support for custom named color replacement with DeviceN color spaces.

Support for different black point compensations.

Lazy initializations for default ICC profiles.

Upgrade of separation devices (speed improvements and no longer limited on number of colors supported).

Ghostscript Color Flow



ICC Profile

Color Management Module (CMM)

ICC Profile

Gray
RGB
CMYK
N-Channel
Named Color

Linked transform from source to device color

Device Colors

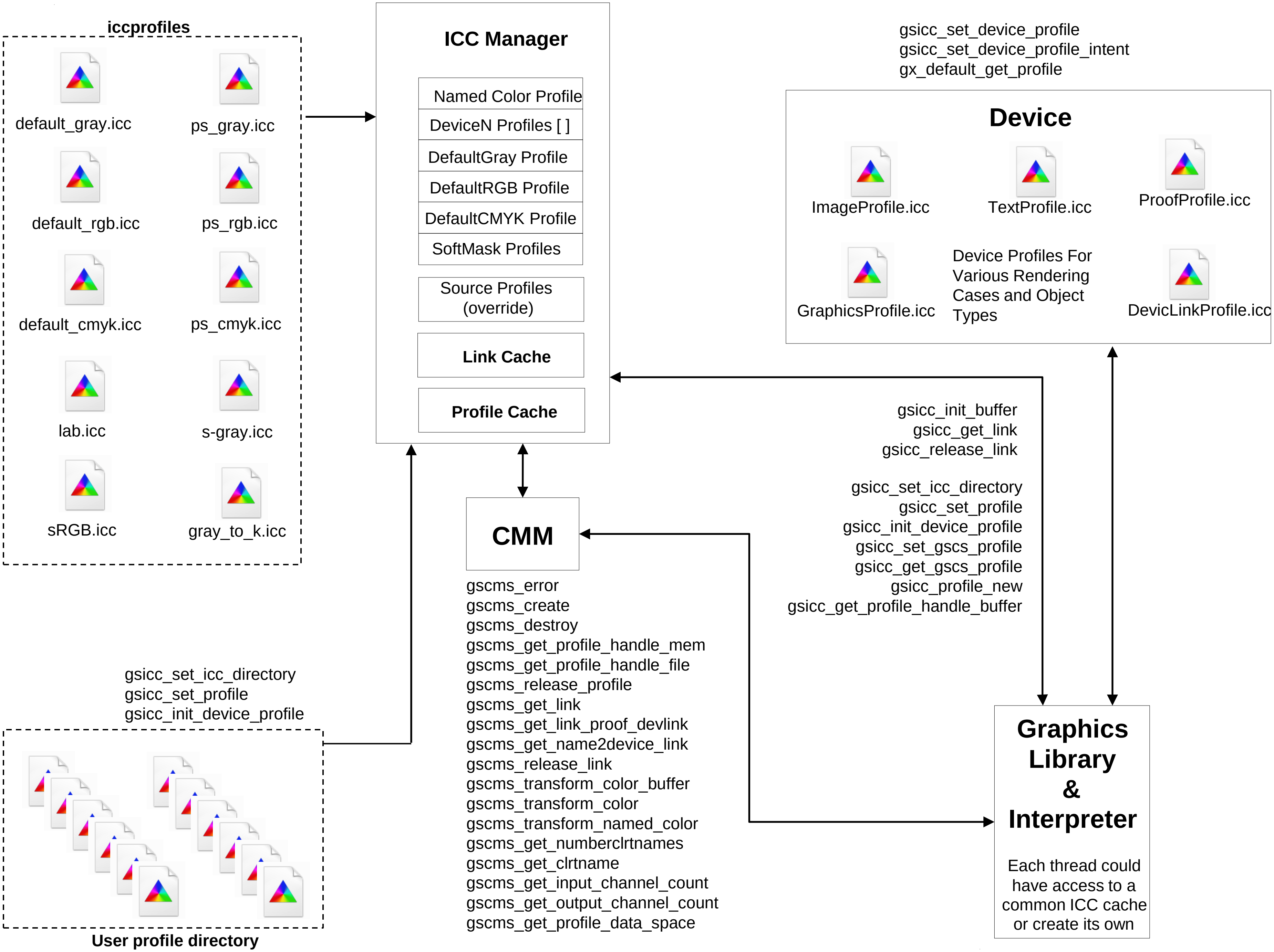
Device Colors

Ghostscript Color Architecture

- Easy to interface different CMM with Ghostscript.
- ALL color spaces defined in terms of ICC profiles.
- Linked transformations and internally generated profiles cached.
- Easily accessed manager for ICC profiles.
- Easy to specify default profiles for DeviceGray, DeviceRGB and DeviceCMYK.
- Devices communicate their ICC profiles and have their ICC profile set.
- Operates efficiently in a multithreaded environment.
- Handles named colors with ICC named color profile or proprietary format.
- ICC Color management of Device-N colors or customizable spot handling.
- Includes object type (e.g. image, graphic, text) and rendering intent into the computation of the linked transform

Ghostscript Color Architecture

- Ability to override document embedded ICC profiles with Ghostscript's default ICC profiles.
- Easy to specify unique **source** ICC profiles to use with CMYK and RGB graphic, image and text objects.
- Easy to specify unique **destination** ICC profiles to use with graphic, image and text objects.
- Easy to specify different rendering intents (perceptual, colorimetric, saturation, absolute colorimetric) for graphic, image and text objects.
- Control to force gray source colors to black ink only for devices that support black ink (e.g. CMYK).



iccprofiles

default_gray.icc
ps_gray.icc
default_rgb.icc
ps_rgb.icc
default_cmyk.icc
ps_cmyk.icc
lab.icc
s-gray.icc
sRGB.icc
gray_to_k.icc

ICC Manager

Named Color Profile
DeviceN Profiles [ ]
DefaultGray Profile
DefaultRGB Profile
DefaultCMYK Profile
SoftMask Profiles
Source Profiles (override)
Link Cache
Profile Cache

gsicc_set_device_profile
gsicc_set_device_profile_intent
gx_default_get_profile

Device

ImageProfile.icc
TextProfile.icc
ProofProfile.icc
GraphicsProfile.icc
Device Profiles For Various Rendering Cases and Object Types
DevicLinkProfile.icc

gsicc_init_buffer
gsicc_get_link
gsicc_release_link

gsicc_set_icc_directory
gsicc_set_profile
gsicc_init_device_profile
gsicc_set_gscs_profile
gsicc_get_gscs_profile
gsicc_profile_new
gsicc_get_profile_handle_buffer

CMM

gscms_error
gscms_create
gscms_destroy
gscms_get_profile_handle_mem
gscms_get_profile_handle_file
gscms_release_profile
gscms_get_link
gscms_get_link_proof_devlink
gscms_get_name2device_link
gscms_release_link
gscms_transform_color_buffer
gscms_transform_color
gscms_transform_named_color
gscms_get_numberclrtnames
gscms_get_clrtname
gscms_get_input_channel_count
gscms_get_output_channel_count
gscms_get_profile_data_space

gsicc_set_icc_directory
gsicc_set_profile
gsicc_init_device_profile

User profile directory

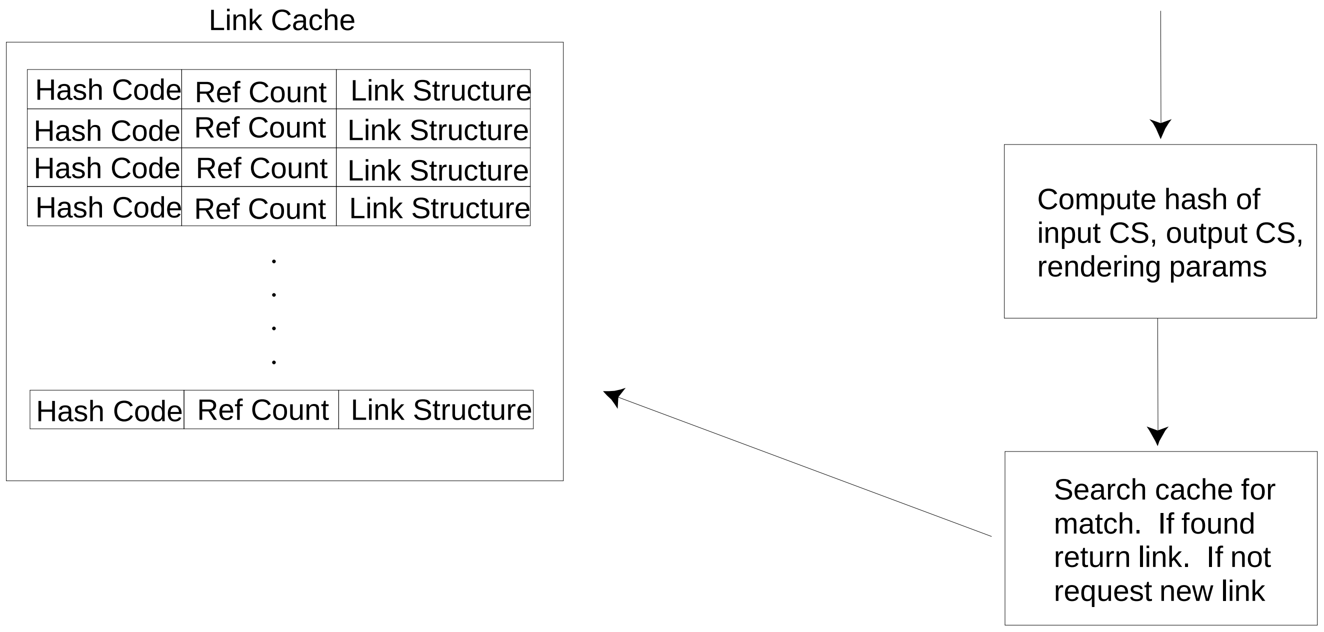
Graphics Library & Interpreter

Each thread could have access to a common ICC cache or create its own

Link Cache

GRAPHICS LIBRARY

gsicc_get_link(* pis, *input_colorspace, *output_colorspace, *rendering_params, memory, include_softproof)



Link Cache

Hash Code	Ref Count	Link Structure
Hash Code	Ref Count	Link Structure
Hash Code	Ref Count	Link Structure
Hash Code	Ref Count	Link Structure
	⋮	
Hash Code	Ref Count	Link Structure

Compute hash of input CS, output CS, rendering params

Search cache for match. If found return link. If not request new link

Link entries are reference counted.

Links are only released if we are at maximum number (or memory), new request is made and a Ref Count is one.

Named Colors

	Required	Optional
Pantone Uncoated Yellow	CIELAB	Device Value
Toyo Red	CIELAB	Device Value
Pantone Coated Green	CIELAB	Device Value
	. . .	
Toyo Coated Blue	CIELAB	Device Value

Missing from ICC profile is ability to use tint information. We provide opportunity for CMM to use. If it cannot, then alternate tint transform is used.

A look-up-table.

There is an ICC profile format for named colors.

In many applications, a custom format is used.

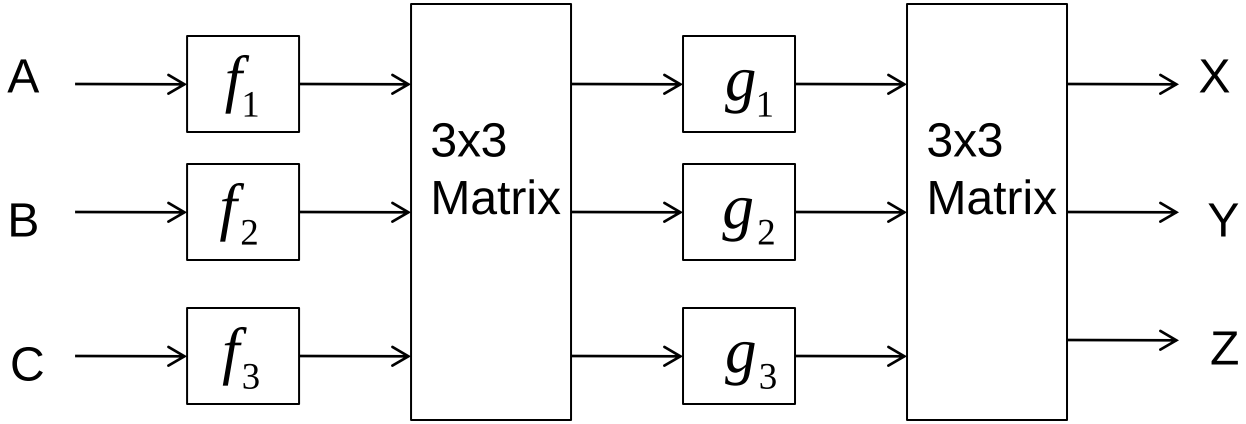
For some companies this is their value added.

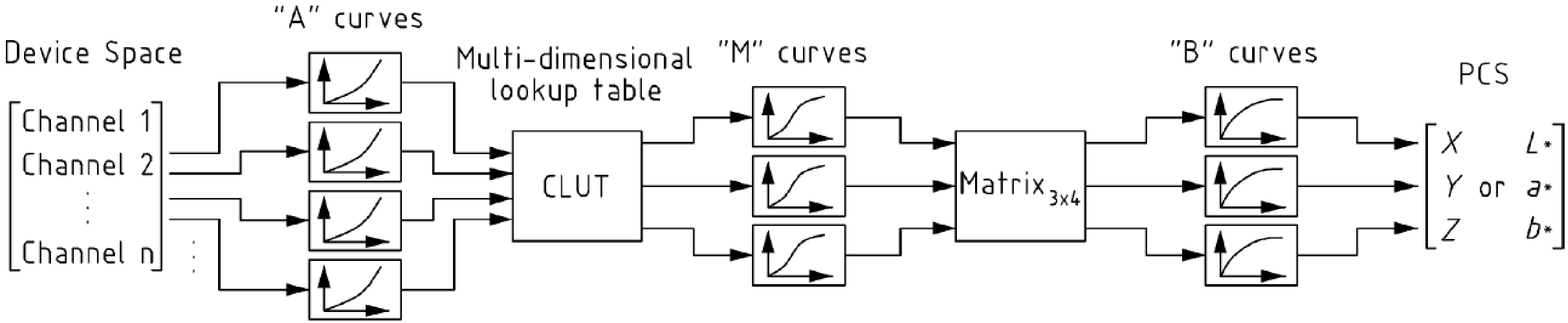
```
int gscms_transform_named_color(gsicc_link_t *icclink,
                                    float tint_value,
                              const char *ColorName,
                           gx_color_value device_values[]);
```

Conversion of PS and PDF Color Spaces

- PS and PDF CIE color spaces are converted to ICC forms that the CMM can handle.
- PS mappings are all 1-way. Device to CIEXYZ or CIEXYZ to Device.
- Procedural mappings are sampled.
- Because of the multiple matrix operations and procedural mappings, some PS color spaces that do not include MLUTs will give rise to ICC profiles that do include MLUTs.

Example PS CIEABC





Profile Cache

- Ghostscript creates ICC profiles from PDF and PS CIE colorspace definitions (e.g. CalRGB, CIEABC, CIEDEFG)
- To avoid repeated creations, these profiles are cached based upon a hash code that is related to the resource ID.
- Cache is designed such that MRU item is at the top of the list.
- Profiles are only released if we are at maximum number (or memory), new request is made and a reference count is one.

Device N color spaces (PDF and PS)

- For Device N output, very simple to provide capability for N-color ICC profile.

- Many desire to have CM with CMYK and to pass additional spot colors unmolested.

- For DeviceN input color, XPS requires ICC profile. PDF and PS use an alternate tint transform.

- Architecture provides capability to define N-color ICC profile for DeviceN input colors to replace the alternate tint transform if desired.

Current Color Command Line Interface

Source Default Profiles

-sDefaultGrayProfile	=	my_gray_profile.icc
-sDefaultRGBProfile	=	my_rgb_profile.icc
-sDefaultCMYKProfile	=	my_cmyk_profile.icc
-sDeviceNProfile	=	my_devicen.icc
-sNamedProfile	=	my_namedcolor_profile.icc

Device Profile

-sOutputICCProfile	=	my_device_profile.icc

ICC Search Directory

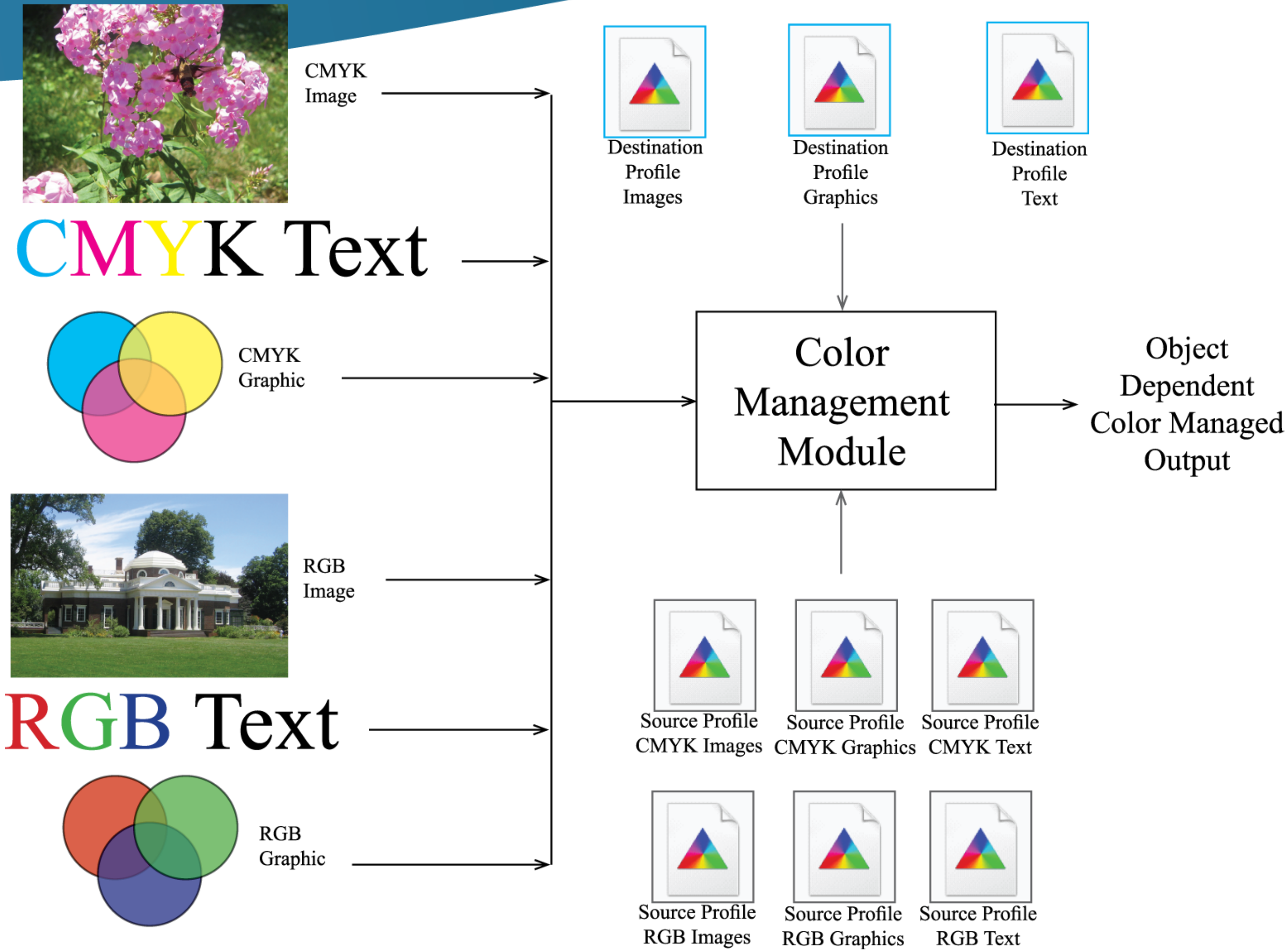
-sICCProfilesDir	=	c:/my_iccprofiles/

Current Color Command Line Interface

Other Settings

-sProofProfile	= my_proof_profile.icc
-sDeviceLinkProfile	= my_link_profile.icc
-dRenderIntent	= intent (0, 1, 2, 3)
-dOverrideICC	= true/false
-dDeviceGrayToK	= true/false
-dUseFastColor	= true/false

Object Dependent Color Management



CMYK Image

CMYK Text

CMYK Graphic

RGB Image

RGB Text

RGB Graphic

Destination Profile Images

Destination Profile Graphics

Destination Profile Text

Color Management Module

Object Dependent Color Managed Output

Source Profile CMYK Images

Source Profile CMYK Graphics

Source Profile CMYK Text

Source Profile RGB Images

Source Profile RGB Graphics

Source Profile RGB Text

Object Dependent Color Management
Source Profiles

Source object dependent control achieved through the command line Specification:

```
-sSourceObjectICC          =    filename
```

Contents of this file define what source profiles should be used with what objects

Key	Profile	Intent
Graphic CMYK	cmyk_src_graphic.icc	0
Image CMYK	cmyk_src_image.icc	0
Text CMYK	cmyk_src_text.icc	0
Graphic RGB	rgb_source_graphic.icc	0
Image RGB	rgb_source_image.icc	0
Text RGB	rgb_source_text.icc	0

Object Dependent Color Management
Destination Profiles

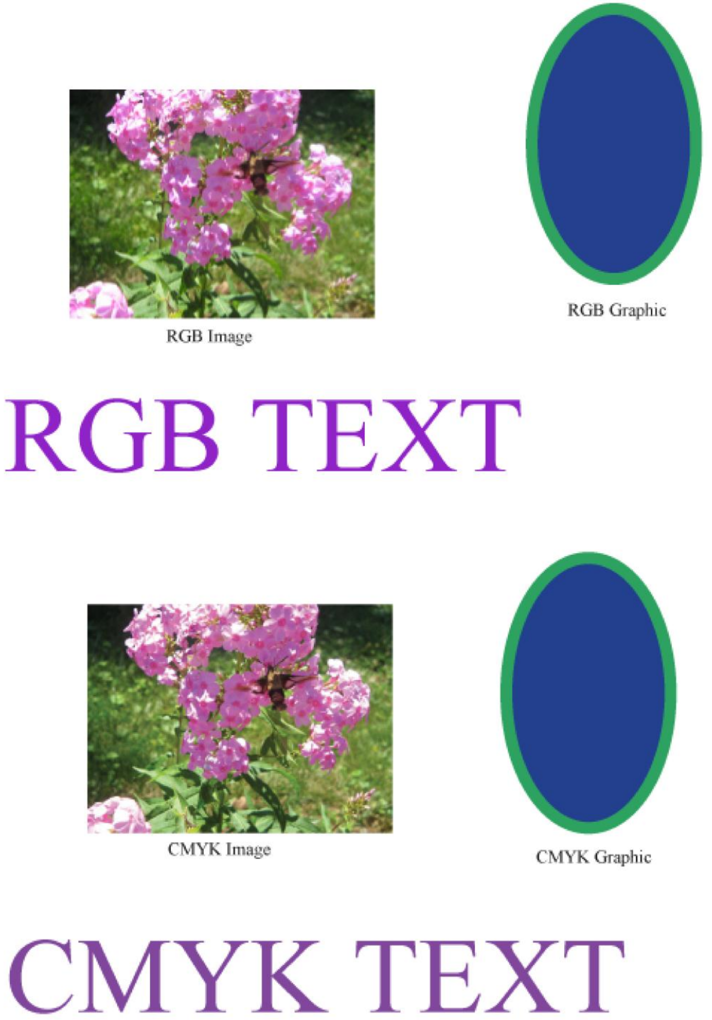
Destination object dependent control achieved through the command line

```
-sTextICCProfile        =   my_device_text_profile.icc
-sGraphicICCProfile     =   my_device_graphic_profile.icc
-sImageICCProfile       =   my_device_image_profile.icc

-sTextIntent            =   intent
-sGraphicIntent         =   intent
-sImageIntent           =   intent
```

Example: Object Dependent CM Default Profiles

Source file includes RGB and CMYK Images, graphics and text.



RGB Image

RGB Graphic

RGB TEXT

CMYK Image

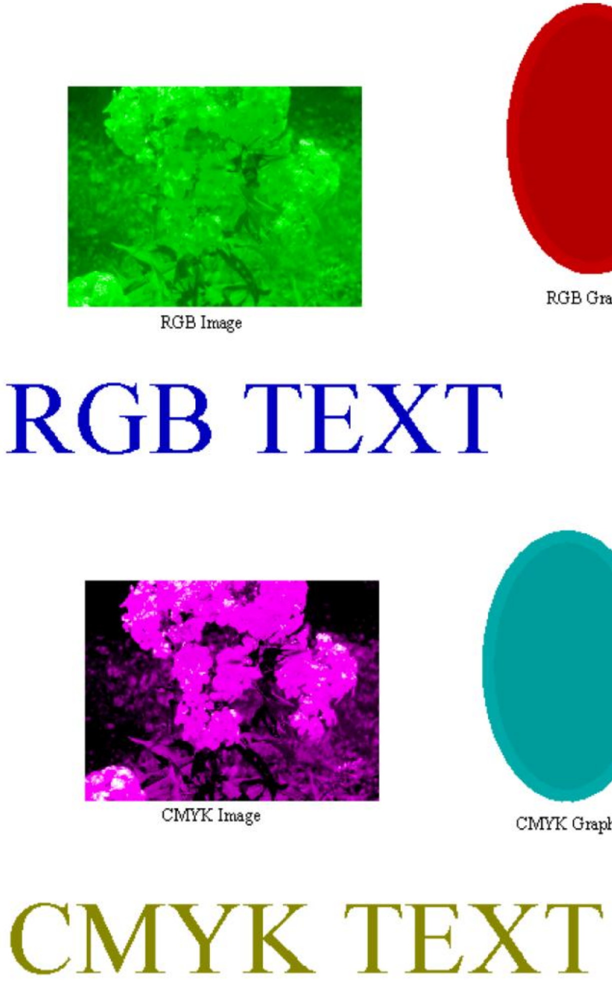
CMYK Graphic

CMYK TEXT

Example: Object Dependent CM Source Profiles Vary

In this case, different ICC profiles were specified to be used with RGB and CMYK graphic, image, and text objects via the Ghostscript command line with –sSourceObjectICC = filename.

Graphic CMYK	cmyk_src_cyan.icc	0
Image CMYK	cmyk_src_magenta.icc	0
Text CMYK	cmyk_src_yellow.icc	0
Graphic RGB	rgb_source_red.icc	0
Image RGB	rgb_source_green.icc	0
Text RGB	rgb_source_blue.icc	0

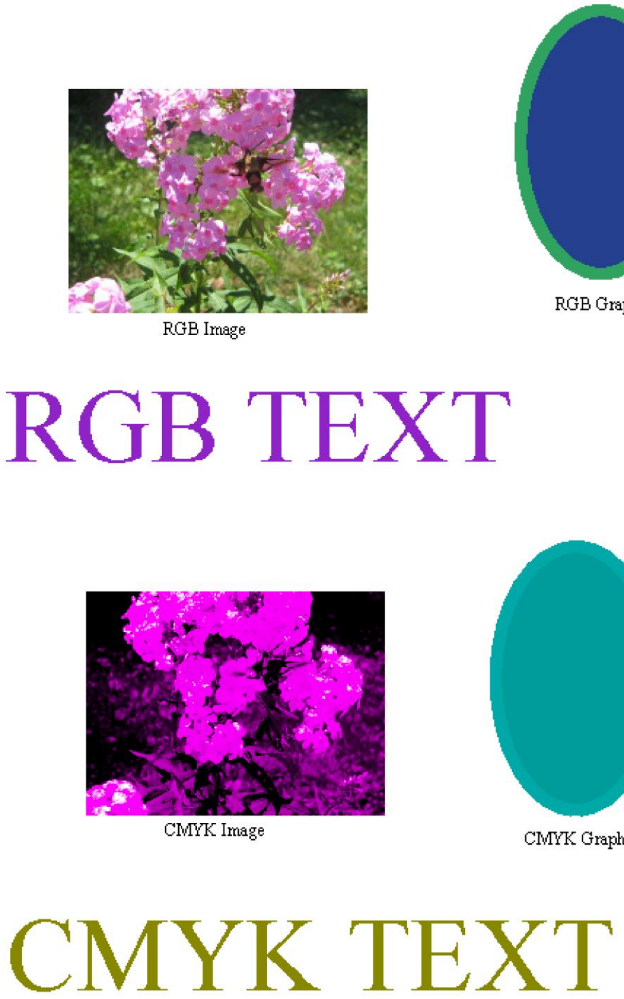


Example: Object Dependent CM
Source CMYK rendering intent varies

In this case, a special source ICC profile for CMYK objects was specified via the Ghostscript command line. The profile was deigned to give radically different results in different rendering intents.

Different rendering intents used for CMYK graphics, images and text

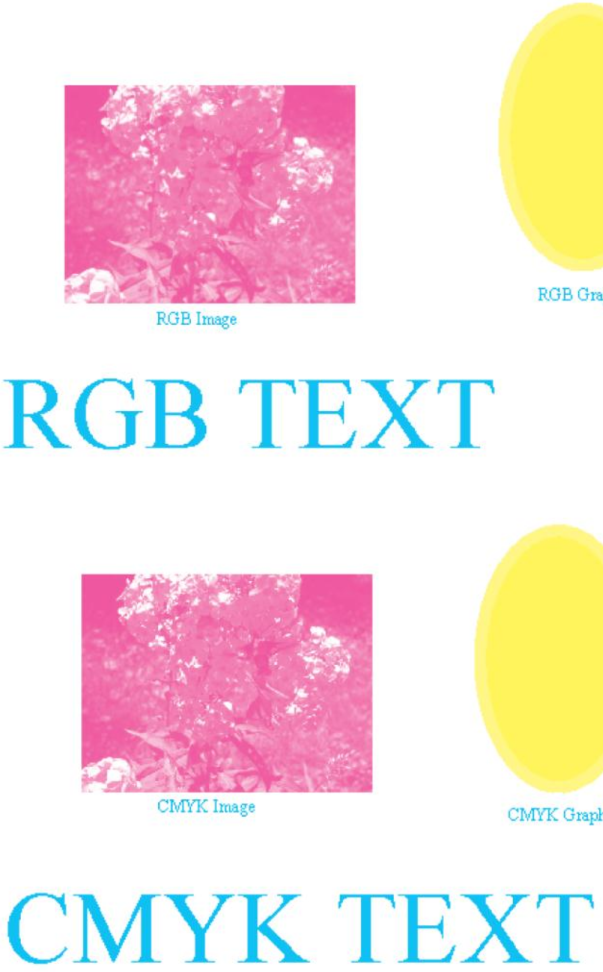
Graphic CMYK	cmyk_src_renderintent.icc	0
Image CMYK	cmyk_src_renderintent.icc	1
Text CMYK	cmyk_src_renderintent.icc	2



Example: Object Dependent CM
Destination Profile varies

Different destination profiles specified for different objects

```
-sGraphicICCProfile  = yellow_output.icc
-sImageICCProfile    = magenta_output.icc
-sTextICCProfile     = cyan_output.icc
```

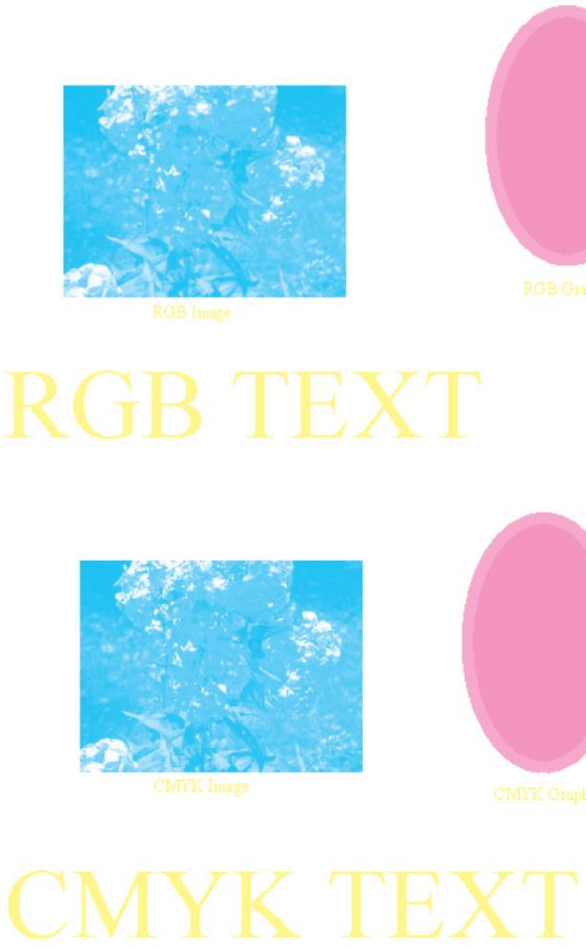


Example: Object Dependent CM
Destination Intent varies

In this case, a special source ICC profile for CMYK objects was specified via the Ghostscript command line.

Different rendering intents used for graphics, images and text

```
-sGraphicICCProfile  = cmyk_des_renderintent.icc
-sImageICCProfile    = cmyk_des_renderintent.icc
-sTextICCProfile     = cmyk_des_renderintent.icc
-dImageIntent        = 0
-dGraphicIntent      = 1
-dTextIntent         = 2
-dOverrideRI
```



Proof and DeviceLink ICC Profile Useage

Two situations:

1) Can I print (or display) on device B what my output will look like if I were to print on device A?

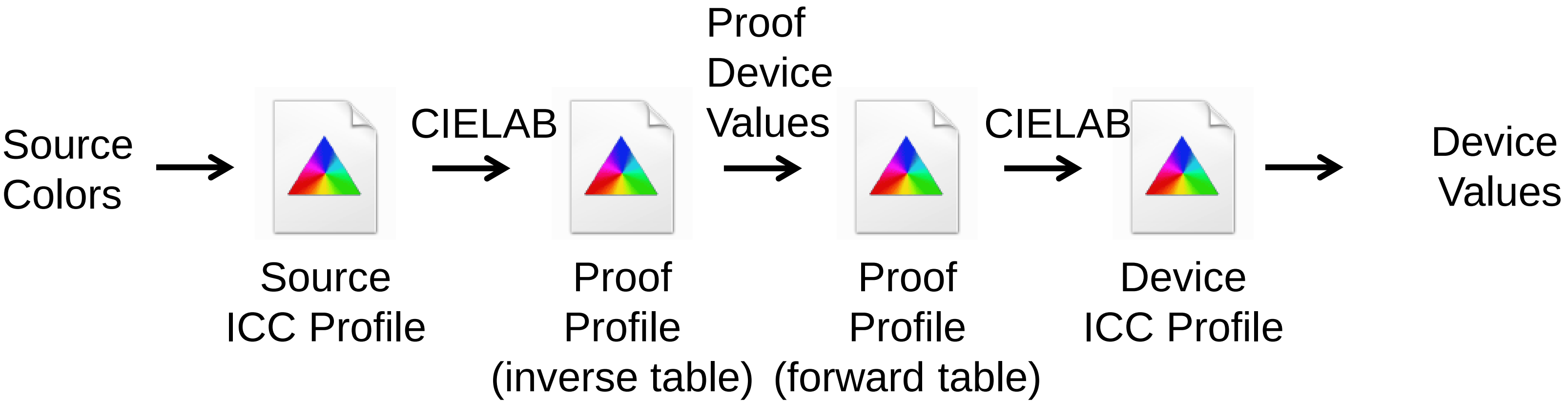
Use a proofing profile.

2) Can I map my output to a common standard space (e.g. Forgra39) and then perform a device link transform to my actual device values?

Use a device-link profile.

Proof and DeviceLink ICC Profile Useage

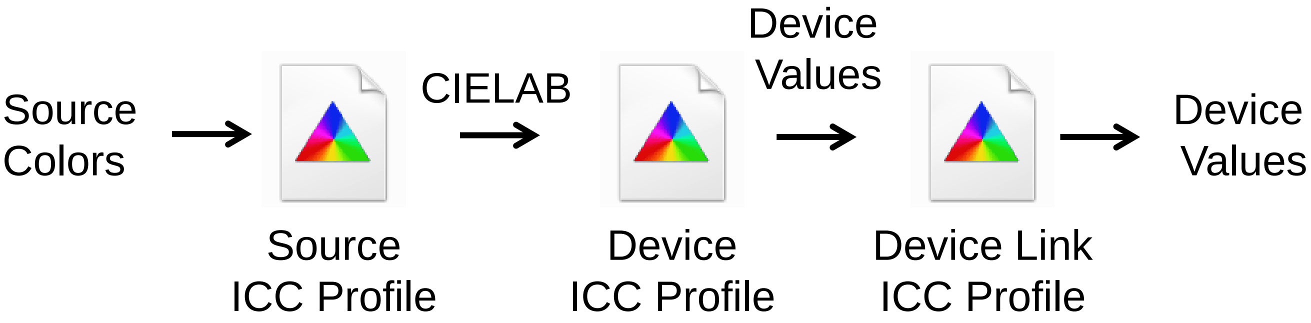
Proof Profile Only Case:



Source Colors → Source ICC Profile → CIELAB → Proof Profile (inverse table) → Proof Device Values → Proof Profile (forward table) → CIELAB → Device ICC Profile → Device Values

Proof and DeviceLink ICC Profile Useage

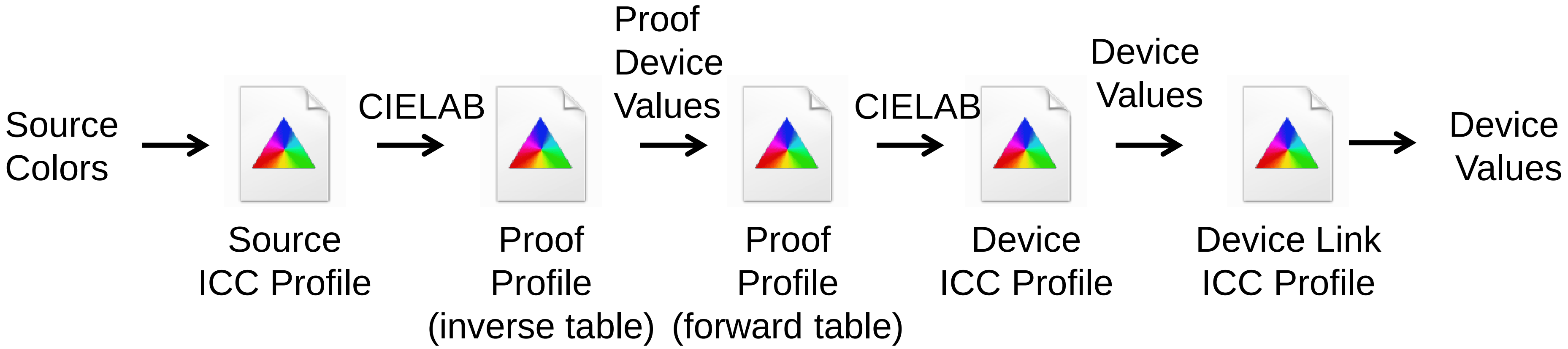
Device Link Profile Only Case:



Source Colors → Source ICC Profile → CIELAB → Device ICC Profile → Device Values → Device Link ICC Profile → Device Values

Proof and DeviceLink ICC Profile Useage

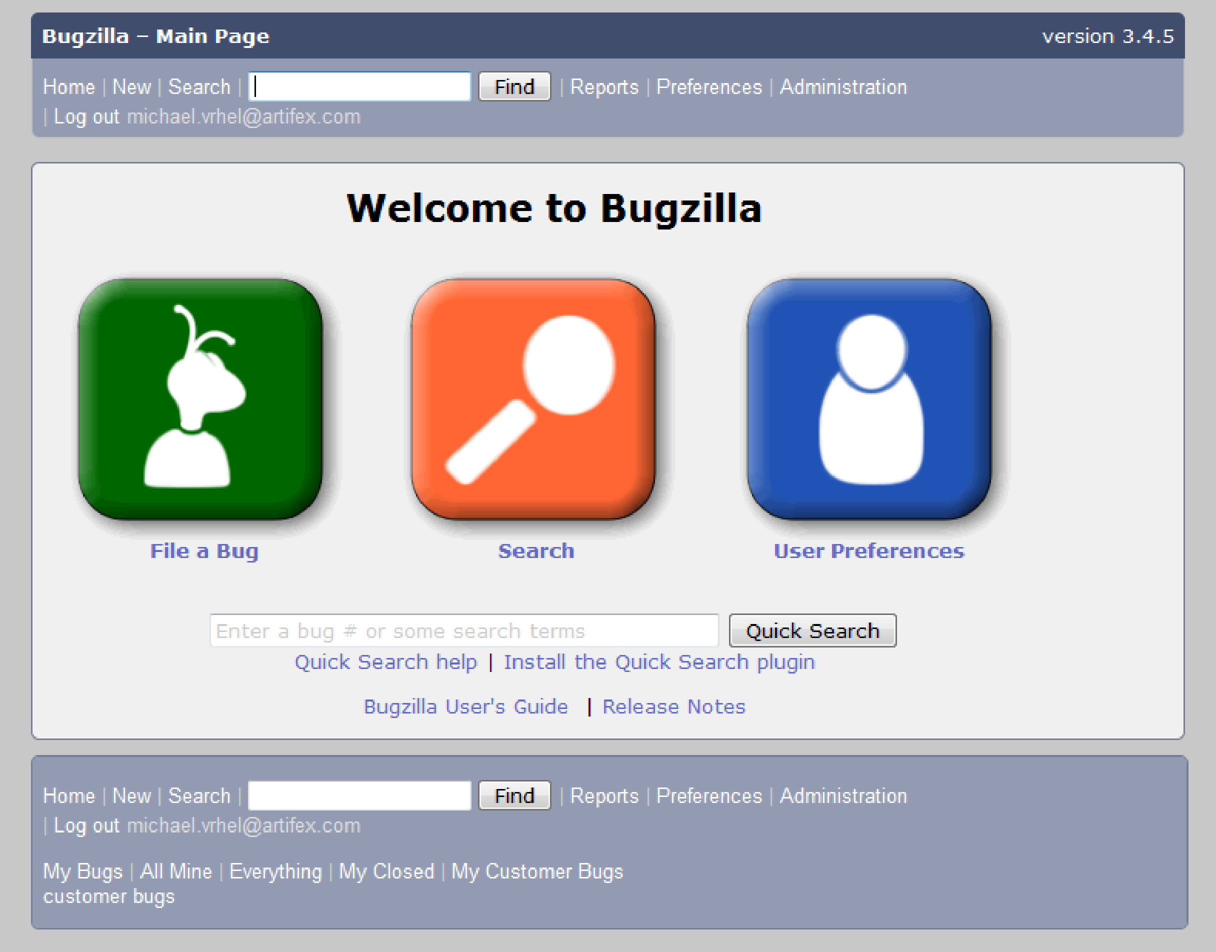
Both proofing and device-link profile.



Source Colors → Source ICC Profile → CIELAB → Proof Profile (inverse table) → Proof Device Values → Proof Profile (forward table) → CIELAB → Device ICC Profile → Device Values → Device Link ICC Profile → Device Values

Bug Tracking

http://bugs.ghostscript.com/



Bugzilla – Main Page version 3.4.5

Home | New | Search | Find | Reports | Preferences | Administration | Log out michael.vrhel@artifex.com

Welcome to Bugzilla

File a Bug

Search

User Preferences

Enter a bug # or some search terms Quick Search

Quick Search help | Install the Quick Search plugin

Bugzilla User's Guide | Release Notes

Home | New | Search | Find | Reports | Preferences | Administration | Log out michael.vrhel@artifex.com

My Bugs | All Mine | Everything | My Closed | My Customer Bugs

customer bugs

No Significant CUPs device issues

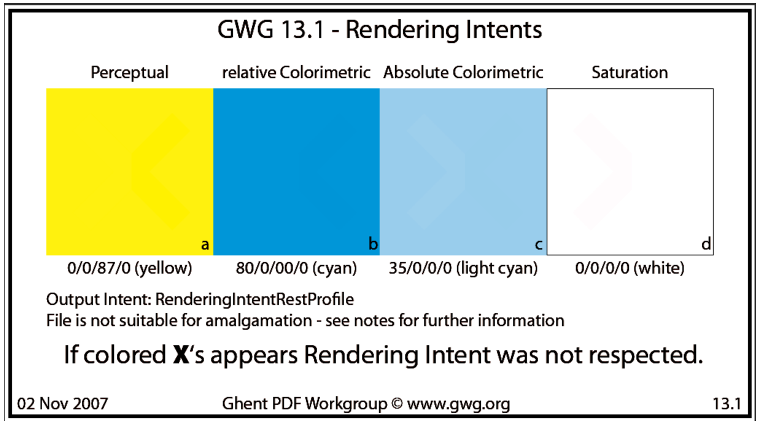
Issues with RGBW color space resolved

Transparency Pattern Color Spaces From CarioGraphics.   Speed issue resolved.

Poor PDF creation from Cario seems to have also been resolved.

PDF Output Rendering Intent

Discussions on OpenICC list about ghostscript NOT supporting the output intent. (Bug 691952)



GWG 13.1 - Rendering Intents

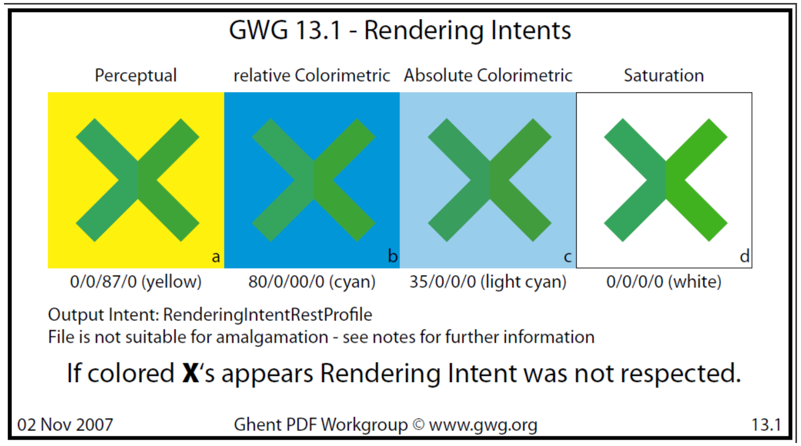
Perceptual relative Colorimetric Absolute Colorimetric Saturation

0/0/87/0 (yellow) 80/0/00/0 (cyan) 35/0/0/0 (light cyan) 0/0/0/0 (white)

Output Intent: RenderingIntentRestProfile
File is not suitable for amalgamation - see notes for further information

If colored **X**'s appears Rendering Intent was not respected.

02 Nov 2007 Ghent PDF Workgroup © www.gwg.org 13.1



GWG 13.1 - Rendering Intents

Perceptual relative Colorimetric Absolute Colorimetric Saturation

0/0/87/0 (yellow) 80/0/00/0 (cyan) 35/0/0/0 (light cyan) 0/0/0/0 (white)

Output Intent: RenderingIntentRestProfile
File is not suitable for amalgamation - see notes for further information

If colored **X**'s appears Rendering Intent was not respected.

02 Nov 2007 Ghent PDF Workgroup © www.gwg.org 13.1

OutputIntents array *(Optional; PDF 1.4) An array of output intent dictionaries describing the color characteristics of output devices on which the document might be rendered (see "Output Intents" on page 970).*

PDF Output Rendering Intent

```
<< /Type /OutputIntent
     /S /GTS_PDFX
     /OutputCondition ( CGATS TR 001 ( SWOP ) )
     /OutputConditionIdentifier ( CGATS TR 001 )
     /RegistryName ( http://www.color.org )
     /DestOutputProfile 100 0 R
>>
```

Per PDF Specification:
DestOutputProfile *(Required if **OutputConditionIdentifier** does not specify a standard production condition; optional otherwise)*

PDF Output Rendering Intent

That leaves us with three issues:

1) If multiple rendering intents are present, which one do we use?

2) If DestOutputProfile entry is not present, what should we do?

3) If output rendering intent ICC profile does not match the process color model of the output device how is the profile used?

PDF Output Rendering Intent

Solution is to introduce a new command line option:

-dUsePDFX3Profile = #

Where # defines which output intent to use in the order that they occur in the document.   If no number specified, first one encountered is used.

If no profile is present in the intent dictionary, a warning is displayed and the rendering intent is ignored.

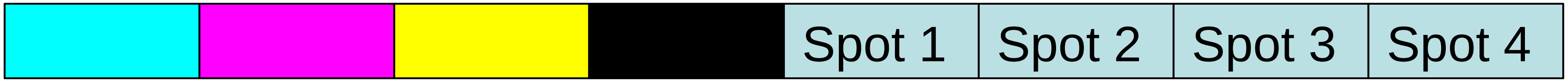
If the output intent ICC profile does not match the process color model of the output device, then the output intent ICC profile is used as a proofing profile.

Planar Separation Devices

Most devices make use of a memory device to buffer the rendered page.

Until recently, Ghostscript was primarily set up for use with chunky memory with the largest chunky pixel being 64 bits.

This presented a limitation for Separation devices with a large number of spot colors.



Spot 1 | Spot 2 | Spot 3 | Spot 4

64 bit word with CMYK + 4 spots

Planar Separation Devices

One approach to solve this was to use a compressed color encoding scheme.

Since certain combinations are more likely to occur.

For example a pure 100% spot with no other colorants is going to be common in label printing.

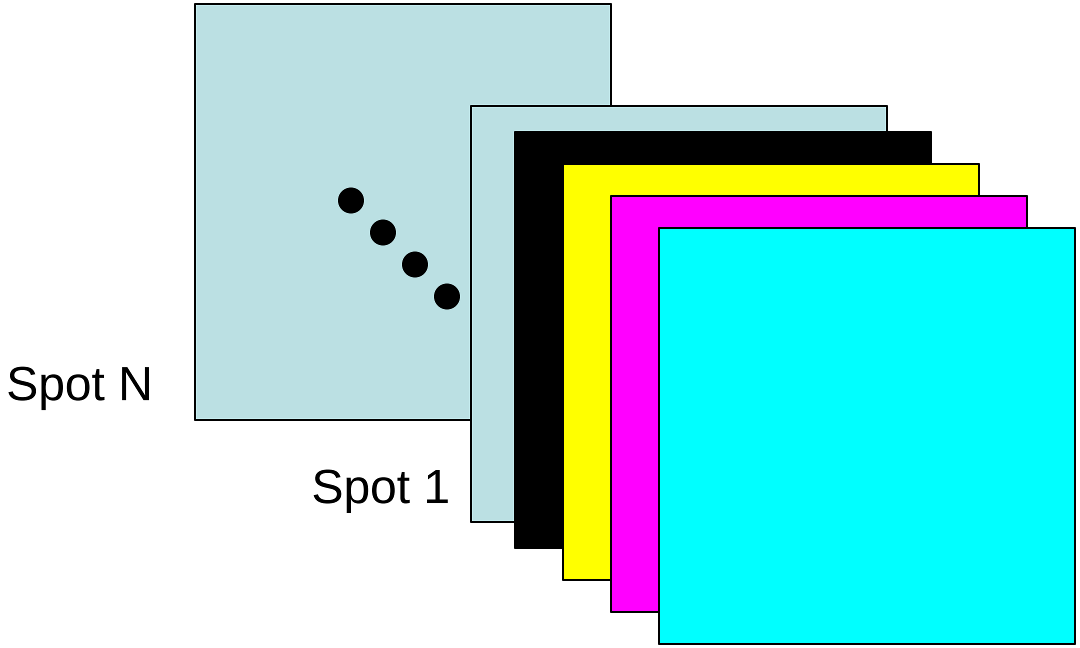
In the presence of transparency and shadings **with** multiple colorants this approach begins to break down.

Planar Separation Devices

Solution is to go to a planar memory memory model.

Advantage for laser print applications

tiffsep and **psdcmyk** devices will make use of this in 9.06 release



Thank you for your attention!

michael.vrhel (at) artifex.com